Noah Buchanan

Problem Set 1
Distributed Systems

September 8, 2020

Functionality that does not work:

As far as I have tested, everything works.

Computational Complexity

1. printAllFiles = $O(n^2)$ I wasnt sure if you wanted us to actually make files and write to them or just add a field to UAFile. This could be to to linear time if that was the case
2. appendToEachFile = $O(n)$
3. removeFileFromCategory = $O(1)$
4. getCategories = $O(n)$
5. listFilesByCategory = $O(n)$
6. addFileToCategory = $O(1)$
7. createCategory = $O(1)$
8. deleteCategory = $O(1)$
9. editCategory = $O(1)$
10. index = $O(n^3)$ reasoning being, the worst case is a Directory that is full of subdirectories, and those subdirectories are full of lists of subdirectories and in each subdirectory as you go down there is also a list of files to go with each subdirectory, therefore 3 different loops you must make, one through the initial directory, one through all the subdirectories in each subdirectory of the main, and one loop through all the files in each subdirectory. However this is extremely theoretical I could not think of any reason from someone to have a file structure like this.

11. listCategories = $O(n)$
12. listAll = $O(nlgn)$

UAFile class

```
import java.io.File;
import java.util.HashMap;

/**
 * Each type of UAFile maintains fileName, the name of the file, pathToFile, the path of
 * associatedCategories, which is HashMap of type <String, UACategory> of associated cat
 * this file, essentially which categories is this file a part of
 * @author noah_
 *
 */

/*********************************
Name: Noah Buchanan
Username: dist103
Problem Set: PS1
Due Date: September 8, 2020
*********************************/

public class UAFile {

        private String fileName;
        private String pathToFile;

        private HashMap<String,UACategory> associatedCategories = new HashMap<>();

        public String getFileName() {
                return fileName;
        }
        public void setFileName(String fileName) {
                this.fileName = fileName;
        }
        public String getPathToFile() {
                return pathToFile;
        }
        public void setPathToFile(String pathToFile) {
                this.pathToFile = pathToFile;
        }

        public UAFile() {
```

```
        }

        public UAFile(File file) {
                this.fileName = file.getName();
                this.pathToFile = file.getAbsolutePath();
        }

        public HashMap<String,UACategory> getAssociatedCategories(){
                return associatedCategories;
        }

}
```

UACategory Class

```
import java.util.HashMap;

/**
 * Contains the fields category name to maintain order of categories and a HashMap of ty
 * of all associated files with this category, or better worded, files CONTAINED in this
 * @author noah_
 *
 */

/*********************************
Name: Noah Buchanan
Username: dist103
Problem Set: PS1
Due Date: September 8, 2020
*********************************/

public class UACategory {

        private String categoryName;

        private HashMap<String,UAFile> associatedFiles = new HashMap<>();

        public String getCategoryName() {
                return categoryName;
        }

        public void setCategoryName(String categoryName) {
                this.categoryName = categoryName;
        }
```

```
        public UACategory() {

        }

        public UACategory(String name) {
                this.categoryName = name;
        }

        public HashMap<String,UAFile> getAssociatedFiles(){
                return associatedFiles;
        }

}
```

UAInvalidFileException Class

```
public class UAInvalidFileException extends Exception {


        /**
         *
         */
        private static final long serialVersionUID = 1L;

        public UAInvalidFileException(){
                super("Invalid File");
        }
}
```

UAInvalidCategoryException Class

```
public class UAInvalidCategoryException extends Exception {

        /**
         *
         */
        private static final long serialVersionUID = 1L;

        public UAInvalidCategoryException() {
                super("Invalid Category");
        }
```

```
}
```

UACategoryExistsException Class

```
public class UACategoryExistsException extends Exception {

        /**
         *
         */
        private static final long serialVersionUID = 1L;

        public UACategoryExistsException() {
                super("Category does not exist");
        }
}
```

UANotEmpyException

```
public class UANotEmptyException extends Exception {

        /**
         *
         */
        private static final long serialVersionUID = 1L;

        public UANotEmptyException() {
                super("Category not empty");
        }
}
```

LionFSManger Class

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;

/**
 * File management system utilizing all the appropriate methods needed for such activiti
```

```
 *
 * @author noah_
 *
 */

/*********************************
Name: Noah Buchanan
Username: dist103
Problem Set: PS1
Due Date: September 8, 2020
*********************************/

public class LionsFSManager {


        public HashMap<String,UAFile> Files = new HashMap<>();
        public HashMap<String,UACategory> Categories = new HashMap<>();
        int task = 1;

        /**
         *
         * @param args reads in one String from the command line, the path of a directory
         * @throws IOException
         * @throws UAInvalidFileException
         * @throws UAInvalidCategoryException
         * @throws UACategoryExistsException
         * @throws UANotEmptyException
         */

        public static void main(String[] args)throws IOException, UAInvalidFileException

                LionsFSManager fileSystem = new LionsFSManager();
                fileSystem.createCategory("nocategory");
                for(UAFile i : fileSystem.Files.values()) {
                        fileSystem.addFileToCategory(fileSystem.Files.get(i.getFileName(
                }
                try {
                        //1
                        System.out.println("\nTask 1");
                        System.out.println("----------------------");
                        System.out.println("Indexing given path");

                        fileSystem.index(new File(args[0]).listFiles(), 0);
                        //2
                        fileSystem.task++;
                        System.out.println("\nTask 2");
```

```
System.out.println("-----------------------");

fileSystem.listAll();
//3
fileSystem.task++;
System.out.println("\nTask 3");
System.out.println("-----------------------");

fileSystem.createCategory("gradefiles");
fileSystem.createCategory("datasets");

fileSystem.listCategories();
//4
fileSystem.task++;
System.out.println("\nTask 4");
System.out.println("-----------------------");

fileSystem.addFileToCategory("datagrades.txt", "gradefiles");
fileSystem.addFileToCategory("grades.txt", "gradefiles");
fileSystem.addFileToCategory("distgrades.txt", "gradefiles");
fileSystem.addFileToCategory("prog2grades.txt", "gradefiles");

fileSystem.listFilesByCategory("gradefiles");
//5
fileSystem.task++;
System.out.println("\nTask 5");
System.out.println("-----------------------");

fileSystem.addFileToCategory("dataset1.txt", "datasets");
fileSystem.addFileToCategory("dataset2.txt", "datasets");
fileSystem.addFileToCategory("cars.txt", "datasets");
fileSystem.addFileToCategory("courses.txt", "datasets");

fileSystem.listFilesByCategory("datasets");
//6
fileSystem.task++;
System.out.println("\nTask 6");
System.out.println("-----------------------");

fileSystem.editCategory("gradefiles", "grades");

fileSystem.listFilesByCategory("grades");
//7
fileSystem.task++;
System.out.println("\nTask 7");
System.out.println("-----------------------");
```

```
                fileSystem.deleteCategory("datasets");

                fileSystem.listCategories();
                //8
                fileSystem.task++;
                System.out.println("\nTask 8");
                System.out.println("-----------------------");

                fileSystem.printAllFiles("grades");

                //9
                fileSystem.task++;
                System.out.println("\nTask 9");
                System.out.println("-----------------------");
                System.out.println("Appending to each file");

                fileSystem.appendToEachFile("grades", "amibcupdatestring");

                //10
                fileSystem.task++;
                System.out.println("\nTask 10");
                System.out.println("-----------------------");

                fileSystem.printAllFiles("grades");

        } catch(Exception e) {
                System.out.println("Error processing task " + fileSystem.task);
        }



}

/**
 *
 * @param category Category of files, whom contents we will be listing
 * @return simply returns a concatenated string of the contents of all files lis
 * @throws IOException
 * @throws UAInvalidCategoryException
 */

public String printAllFiles(String category)throws IOException,UAInvalidCategory


        String concat = "";
```

```
        System.out.println("ffdsa");

        for(UAFile j : Categories.get(category).getAssociatedFiles().values()) {
                BufferedReader br = new BufferedReader(new FileReader(j.getPathT
                String line = "";

                while((line = br.readLine()) != null) {
                        System.out.println(j.getFileName() + "'s contents---->" +
                        concat += j.getFileName() + "'s contents---->" + line +

                }

                br.close();
        }


        return concat;
}

/**
 *
 * @param category Category of files, whom we will be appending a given String t
 * @param contentToAppend the content of which we will be appending to each file
 * @return returns previous content of files with the new content appended
 * @throws IOException
 * @throws UAInvalidCategoryException
 */

public String appendToEachFile(String category, String contentToAppend) throws I

        for(UAFile j : Categories.get(category).getAssociatedFiles().values()) {
                FileWriter writer = new FileWriter(j.getPathToFile());
                writer.append(contentToAppend);
                writer.close();
        }

        return contentToAppend;
}

/**
 * removes a given file from a given category
 * @param file file to be removed
 * @param category category the file will be removed from
 * @return returns true/false based on whether the removal was successful
 * @throws UAInvalidCategoryException
 */
```

```
public boolean removeFileFromCategory(String file, String category) throws UAInva

        try {
                int listSize = getCategories(file).size();
                if(listSize == 1 && Categories.get(category).getCategoryName().e
                        Categories.get("nocategory").getAssociatedFiles().put(fi
                }
                if(Categories.get(category).getCategoryName().equals("nocategory
                        System.out.println("cannot remove category association t
                        return false;
                } else {
                        Categories.get(category).getAssociatedFiles().remove(fil
                }
                return true;
        } catch(Exception e) {
                return false;
        }
}

/**
 *
 * @param file given file to find the categories for
 * @return returns the categories a specified file is associated with in an Hashl
 * @throws UAInvalidFileException
 */

public HashMap<String,UACategory> getCategories(String file) throws UAInvalidFil

        HashMap<String,UACategory> list = new HashMap<>();
        for(UACategory j : Files.get(file).getAssociatedCategories().values()) {
                list.put(j.getCategoryName(),j);
                System.out.println(j.getCategoryName());
        }

        return list;
}

/**
 *
 * @param category lists the associated files contained in a category
 * @return returns the associated files contained in the given category in a Hasl
 * @throws UAInvalidCategoryException
 */

public HashMap<String,UAFile> listFilesByCategory(String category) throws UAInval
```

```
        HashMap<String,UAFile> list = new HashMap<>();
        for(UAFile j : Categories.get(category).getAssociatedFiles().values()) {
                list.put(j.getFileName(), j);
                System.out.println(j.getFileName());

        }
        return list;
}

/**
 *
 * @param file file to be added to specified category
 * @param category category specified file will be added to
 * @return returns true/false based on if the file insertion was successful
 * @throws UAInvalidCategoryException
 * @throws UAInvalidFileException
 */

public boolean addFileToCategory(String file, String category)throws UAInvalidCa

        try {

                File path = new File(file);
                UAFile fileToAdd = new UAFile(path);

                if(Categories.get("nocategory").getAssociatedFiles().get(file) ==

                Files.put(fileToAdd.getFileName(), fileToAdd);

                Categories.get(category).getAssociatedFiles().remove(fileToAdd.g

                path.createNewFile();
                fileToAdd.getAssociatedCategories().put(fileToAdd.getFileName(),
                Categories.get(category).getAssociatedFiles().put(fileToAdd.getF
                } else {

                        Categories.get(category).getAssociatedFiles().remove(fil
                        fileToAdd.getAssociatedCategories().put(fileToAdd.getFil
                        Categories.get(category).getAssociatedFiles().put(fileTo
                }

                return true;
        } catch(Exception e){
                return false;
        }
```

```
}

/**
 *
 * @param category given name of category to be created
 * @return returns the Object of the new category created
 */

public UACategory createCategory(String category) {

        UACategory c = new UACategory(category);
        Categories.put(category, c);
        return c;

}

/**
 *
 * @param category given category to be deleted
 * @return returns the Object of the category deleted
 * @throws UACategoryExistsException
 * @throws UANotEmptyException
 */

public UACategory deleteCategory(String category)throws UACategoryExistsExceptio

        return Categories.remove(category);

}

/**
 *
 * @param oldName previous name of category we want to change
 * @param newName name to replace the old category name
 * @return returns the new renamed category
 * @throws UACategoryExistsException
 * @throws UAInvalidCategoryException
 */

public UACategory editCategory(String oldName, String newName)throws UACategoryE


        UACategory temp = Categories.get(oldName);
        Categories.remove(oldName);
        Categories.put(newName, temp);
        return Categories.get(newName);
```

```
}

/**
 *
 * @param list a given list of files to be index
 * @param i i for recursion purposes
 */

public void index(File[] list, int i) {

        if (i >= list.length) {
                return;
        } else {
                if (list[i].isDirectory()) {
                        index(list[i].listFiles(), 0);
                } else {
                        UAFile file = new UAFile();
                        file.setFileName(list[i].getName());
                        file.setPathToFile(list[i].getAbsolutePath());
                        Files.put(file.getFileName(), file);
                }
                index(list, ++i);
        }

}

/**
 *
 */

public void listCategories() {

        for(UACategory i : Categories.values()) {
                System.out.println(i.getCategoryName());
        }
}

/**
 *
 * @return returns the list that it just printed in ascending order, unaltered, a
 * not sure if you wanted us to alter the list or just print it
 */

public HashMap<String,UAFile> listAll() {

        String[] x = new String[Files.size()];
```

```
        int num = 0;
        for(UAFile i : Files.values()) {
                x[num] = Files.get(i.getFileName()).getFileName();
                num++;
        }
        mergeSort(x,0,x.length-1);
        for(int i = 0; i < x.length; i++) {
                System.out.println(x[i]);
        }

        return Files;
}

/**
 *
 * @param A Array to be sorted
 * @param p beginning index of array
 * @param r ending index of array
 */

public static void mergeSort(String[] A, int p, int r) {

        if (p < r) {
                int q = (p + r) / 2;
                mergeSort(A, p, q);
                mergeSort(A, q + 1, r);
                merge(A, p, q, r);
        }
}

/**
 *
 * @param A Array to merge
 * @param p beginning index of array
 * @param q middle index of array
 * @param r ending index of array
 */

public static void merge(String[] A, int p, int q, int r) {

        int n1 = q - p + 1;
        int n2 = r - q;

        String[] L = new String[n1 + 1];
        String[] R = new String[n2 + 1];
```

```
                for (int i = 0; i < n1; i++) {
                        L[i] = A[p + i];
                }
                for (int j = 0; j < n2; j++) {
                        R[j] = A[q + j + 1];
                }

                L[n1] = "zzz";
                R[n2] = "zzz";

                int i = 0, j = 0;

                for (int k = p; k <= r; k++) {

                        if (L[i].compareTo(R[j]) < 0) {
                                A[k] = L[i];
                                i++;
                        } else {
                                A[k] = R[j];
                                j++;
                        }
                }
        }

}
```